AMENDMENTS TO THE SPECIFICATION

Please replace the paragraph on page 5, lines 21-31 of the application as originally filed with the following amended paragraph.

In one embodiment of the invention, ARL Table 5 may be used without a shared memory structure. In this case, it is desirable for the table to be configured as an one-way associative, i.e., direct mapped, memory. In embodiments of the invention in which the ARL Table 5 is shared with Table 4, Table, 6, or both, as well as with pool memory 8, it may be desirable to use another type of memory structure, including, without limitation, an n-way associative memory, a hash table, binary searching structure, and a sequential searching structure. One skilled in the art could readily select the appropriate [[a]] search technique for a given structure.

Please replace the paragraph on page 6, lines 20-29 of the application as originally filed with the following amended paragraph.

By using a preselected portion of the packet destination address as an index into ARL Table 5, [[a]] an address match can be resolved quickly, and the packet passed to the appropriate port for transmission. This destination address key direct-mapped address search enables multi-port packet-based switch 1 to be operable, for example, at wire speed in full-duplex, non-blocked, 100Base-TX operations. One skilled in the art would realize that the contemplated invention can be practiced in environments other than 100Base-T environments and at wire speeds in excess of 100 Mb/s.

Please replace the paragraph on page 9, lines 1-14 of the application as originally filed with the following amended paragraph.

FIG. 4 illustrates one embodiment of the direct-mapped address table indexing using, for example, a 13-bit key derived from the 48-bit MAC address value, i.e., the Ethernet frame destination address field 21. As previously described, the least significant bit 23 of address value 21 is mapped to the least significant bit 24 of key 22. In this example, the address table entries~~, therefore,~~ are offset in the address space from the index by [[$F0_h$]] an offset 29, which is, therefore, included in the key 22. The most significant bit location 25 can obtain its value 26

from bit 35 of the corresponding MAC address value 21. If desired, a fourteenth bit from MAC address 21 can be used to provide a bit value 28 for the most significant bit 27 for key 22.

Please replace the paragraph on page 12, lines 7-21 of the application as originally filed with the following amended paragraph.

In an embodiment of the present invention, a fixed number of buffers are employed. Used buffers are those that have been granted to a receive port, but have not yet been returned, or freed, by a transmit port. All of the remaining buffers are designated "unused". It also is the buffer manager's responsibility [[also]] to track unused buffers so that they can be granted. Although one simple method to track unused buffers is to maintain a buffer list, such a list may create undesirable space limitations on a switch device because the list area must be long enough to store all of the buffers in a system and, further, each location in the list must be able to store the number of, or a pointer to, any buffer in the system. In a device having 512 list locations, for example, with each location having a corresponding nine-bit pointer, 4608 bits of storage would be required.

Please replace the paragraph on page 13, lines 12-13 of the application as originally filed with the following amended paragraph.

1) SEARCH [[(61)]] (62) - search for zero-valued bits that are in the buffer control array, indicating the location of a free buffer;

Please replace the paragraph on page 13, lines 14-16 of the application as originally filed with the following amended paragraph.

2) FREE [[(62)]] (61) - write a zero to a bit location specified by free controller 51, thus freeing the associated buffer for allocation; and

Please replace the paragraph on page 13, lines 17-18 of the application as originally filed with the following amended paragraph.

3) ALLOCATE (63) - write a one value to a bit location that was identified during search state [[61]] 62 by search engine 52.